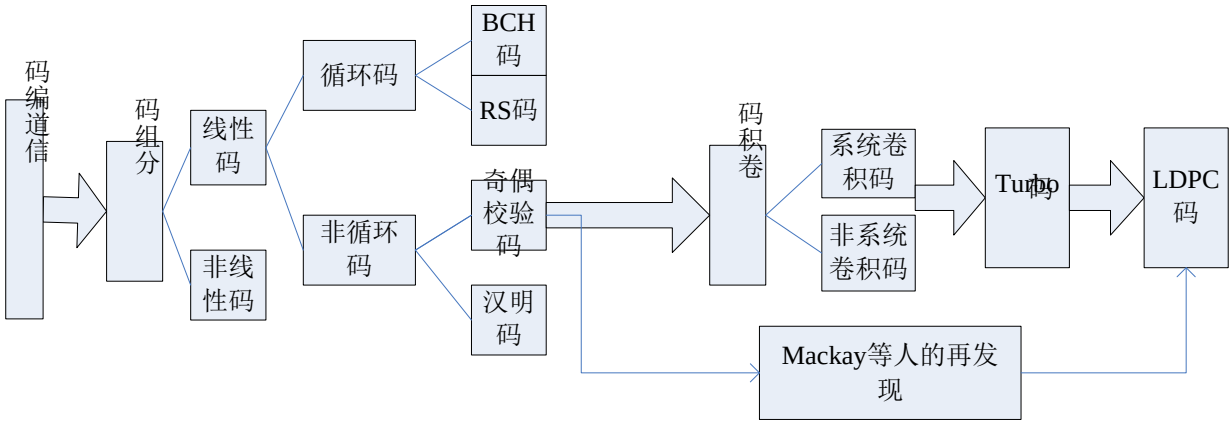
第八章 LDPC 码

8.1 研究 LDPC 码的原因



可以看到，信道编码的发展可以简单的归纳为分组码→卷积码→分组码这样一个过程（在这里按其结构将 Turbo 码也归入卷积码的范畴）。其中 Turbo 码的出现以及迭代译码的思想引入使得信道编解码产生了前所未有的飞跃，但 Turbo 码之后卷积码却没有更大的发展，究其原因就是其没有完备的理论基础，使得人们不能给出其性能上严密的数学解释。于是在那之后可以说是一种退化或者是返朴归真，Mackay 等人再次发掘了 Gallager 于 1962 年提出的一种具有稀疏校验矩阵的分组纠错码，即 LDPC 码。

LDPC 码和传统的分组码最大的区别在于它们的译码。分组码通常是用最大似然译码，因此，码长一般较短，并用代数方法设计使得复杂度较低。而 LDPC 码是迭代译码，校验矩阵用 Tanner 图表示，码长更长、围绕着校验矩阵 H 的特性进行设计是核心。

LDPC 码自身的矩阵结构引入了交织特性，而且其采用迭代译码的

方法，使其性能比以往的线性分组码有很大程度的提高。由于其基本原理是基于最原始的线性分组码，因此它有强大的数学工具作为其理论依据，几乎融图论、组合数学、概率论、矩阵论、代数、几何、代数数论、黎曼几何于一炉。在通信的其它领域，我们很难再找到某个方向可以有如此深厚的理论基础与之媲美。

但理论上的完备性并不能使其直接应用于实际，因此从码字构造的方向来说，如何将 LDPC 码应用于实际工作才是值得深入研究的。为了保证其实现性，性能上就要有所妥协。

在编码方面，以准循环 LDCP 码，即 QC-LDPC 码为例，为了降低硬件上的存储空间以及易于编码，就要以牺牲 LDPC 码先天的优势——交织特性为代价，这样便做出了性能与实现上的折中。但这种折中是有意义的，为 LDPC 码的实际应用开辟了道路。而且，通过某些方法，可以使设计出的码字在易于存储实现的同时，还能保证一定的性能。

在译码方面，种种方法都可以归结为和积算法（SPA）的变形，都是在其基础上做出改进，从而保证译码性能前提下使译码器尽可能的简单。相对于 Turbo 码，LDPC 码的解码迭代次数还是过高，这样在实际应用中的竞争力便大打折扣。于是，怎样在保证性能的前提下降低译码时间是现在译码研究的主要工作。

8.2 LDPC 码简介

1996 年 Mackay、Spielman 和 Wiberg 几乎同时发现：Gallager

早在 1962 年提出的低密度校验码（简称 LDPC 码，也称 Gallager 码）也是一个好码，具有更低的线性译码复杂度。Gallager 提出 LDPC 码后一直没有得到编码界的重视，只有 1981 年 Tanner 从图论的角度研究过 LDPC 码。

自 Mackay 等“再发现”LDPC 码后，人们的进一步研究表明：给予非规则双向图的 LDPC 长码的性能可以优于 Turbo 码，而且这样的码的性能可以非常接近 Shannon 限。一个原因在于 LDPC 码具有良好的距离特性、较小的译码错误概率和较低的译码复杂度，且码长大于 200 时不存在错误平台，码率容易调整，实验结果中几乎均为可检测错误，所以 LDPC 码无论在理论上还是在实际上都具有极其重要的价值。LDPC 码的重新发现是继 Turbo 码后在纠错码领域又一重大进展。

（1）编码方面的简介

无论 Gallager 还是 Mackay 都是用随机的方法构造 LDPC 码，用随机法构造的 LDPC 码的码字参数选择灵活，但对于高码率、中短长度的 LDPC 码用随机法进行构造，要避免短循环是困难的，其没有一定的码结构，编码复杂度高，于是人们考虑用代数法构造 LDPC 码。

LDPC 码代数构造可采用几何方法、图论方法、实验设计方法、置换方法来设计。不同的构造方法都是为了实现以下几个目的：增大图中最小循环长度，即 Girth 值。优化非规则码的节点分布，减小编码复杂度，构造的 LDPC 码要有好的码性能。

M.G.Luby 等指出，非规则 H 矩阵构造的码字性能优于相应的规则 H 矩阵构造的码字。在寻找好的码结构方面，Mackay 等提出：能

快速编码的 LDPC 矩阵通常具有下三角形结构。

T.J.Richardson 探讨了如何构造编码矩阵，使编码时间与码块长度实际上符合线性关系(线形时间编码)，而非通常认为的平方关系。

Y.Kou 和 S.Lin 等探讨了基于有限几何学的 LDPC 码结构。S.Lin 研究团队的 B.Ammar 等提出用均衡不完全区组设计方法（BIBD）构造好的 LDPC 码。

（2）译码方面的简介

Gallager 曾给出两种 LDPC 码的迭代译码算法：硬判决和软判决算法。后者虽有好的性能，但太复杂，消息传递算法可以认为是二者的折衷。消息传递算法（MP，Message Passing）有时也称置信传播（BP，Belief Propagation）算法。在 MP 译码算法中，节点到节点的消息是通过 Tanner 图传递的。

译码算法的改进和优化离不开译码性能的分析。在译码性能分析的研究方面， T.J.Richardson 等开发了一种在码块无限长假设条件下，跟踪 LDPC 码 Tanner 图中消息概率的技术，称为“密度进化”算法的数值程序，来近似估计噪声门限（在该门限以下可望成功地采用 BP 算法），提出了一种通用方法来确定任何二元输入无记忆信道中采用 MP 译码的 LDPC 码的性能。特别对于 BP 译码算法，该方法可提供任何所需精度的性能估计。

S.Y.Chung 等将消息离散化，通过计算机迭代搜索寻找最优的节点次数分布，特别适合于非规则码的分析，在二进制输入 AWGN 信道下，设计码率 1/2、码长 10^7 的非规则 LDPC 码在错误概率 10^{-6} 时离

Shannon 限仅 0.0045dB。这是迄今为止报道出的性能最接近 Shannon 限的信道编码。

8.3 LDPC 码的基本概念

LDPC 码是用一个稀疏的非系统的校验矩阵 H 定义的线性码。

行重：H 矩阵每行中“1”的个数，其值远远小于 H 矩阵的列数。

列重：H 矩阵每列中“1”的个数。

LDPC 码可以按照 H 矩阵分为规则（regular）和非规则（irregular）两种。规则 LDPC 码中，各行的行重是一致的，各列的列重也是一致的，而行重或者列重不一致就称为非规则 LDPC 码。

下面给出规则 LDPC 码的定义：

一个（*n, j, k*）的规则 LDPC 码由它的校验矩阵 H 定义，校验矩阵有 *n* 列，*m* 行，列重 *j*，行重 *k*，其中 $m = n \cdot j / k$，*j<k*，*j<<m*，*k<<n*。

LDPC 码的 H 矩阵一般都是用非系统形式给出的。例如我们生成一个(6,2,4)的校验矩阵：

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

如例所示的规则 H 矩阵行重为 4，列重为 2。

同一般的线性分组码，H 矩阵的码率可以如下计算：*R=(n-m)/n=(k-j)/k*；则图中的矩阵可以用来实现码率 1/2 的编码。

对于一个非正则（irregular）LDPC 码，我们常用度分布（degree distribution）来描述它。H 中，列重为 i 的占比 v_i 和行重为 j 的占比

h_j，则 v=[v_1, v_2,..]和 h=[h_1,h_2,…]就是该码字的度分布。

例如：

$$H=\begin{bmatrix}1&1&0&1&0&0\\1&1&1&0&0&1\\0&1&1&0&1&0\end{bmatrix}$$

该码字的度分布 v_1=1/2、v_2=1/3、v_3=1/6，h_3=2/3、h_4=1/3

$$n\left(\sum_i i\cdot v_i\right)=m\left(\sum_j j\cdot h_j\right)$$

其码率可表示为：

$$r=1-\frac{\sum_i i\cdot v_i}{\sum_j j\cdot h_j}=1-\frac{1*\frac{1}{2}+2*\frac{1}{3}+3*\frac{1}{6}}{3*\frac{2}{3}+4*\frac{1}{3}}=1-\frac{\frac{5}{3}}{\frac{10}{3}}=\frac{1}{2}$$

LDPC 码的校验矩阵的行对应着校验方程（校验节点），列对应着传输的比特（比特节点），它们之间的关系可以用 **Tanner** 图来表示，图的左边有 ***n*** 个节点，每个节点表示码字的信息位，称为比特节点{x_j, j=1, 2, …, n}，对应于校验矩阵的各列，比特节点也称为信息节点或变量节点；右边有 ***m*** 个节点，每个节点表示码字的一个校验集，称为校验节点{r_i, i=1, 2, …, m}，代表校验方程，对应于校验节点的各行；与校验矩阵中“1”元素相对应的左右两节点之间存在连接边。我们将这条边两端的节点称为相邻节点，每个节点相连的边数成为该节点的度（**Degree**），每个比特节点与 ***j*** 个校验节点相连，称为该比

特节点的度为 *j*；每个校验节点与 *k* 个信息节点相连，称为校验节点的度为 *k*。例如上面的 H 矩阵对应的 Tanner 图如下：

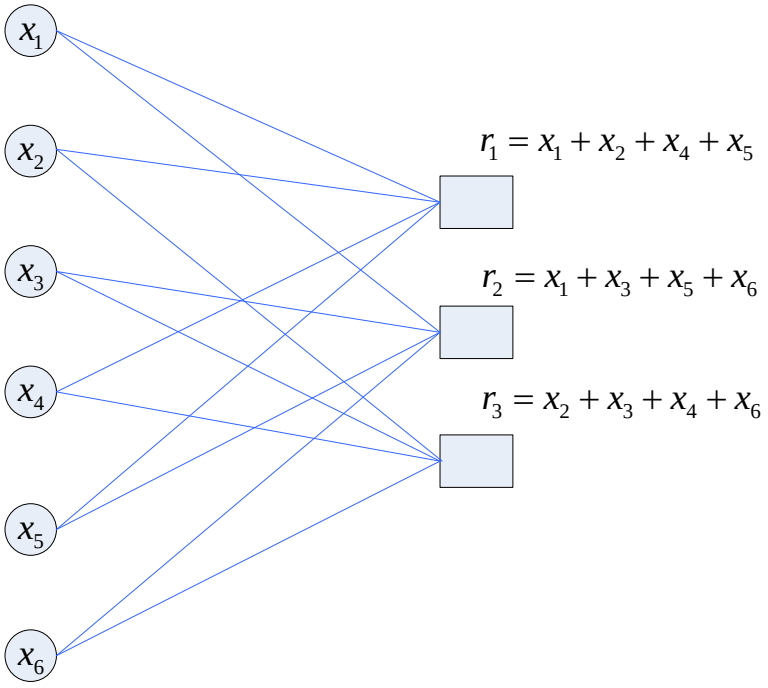


图 1 Tanner 图

LDPC 码校验矩阵 H 中，一个很重要的概念：H 矩阵的最小圈长，即 girth。一个 4 循环（girth=4）在图 1 中表示为：

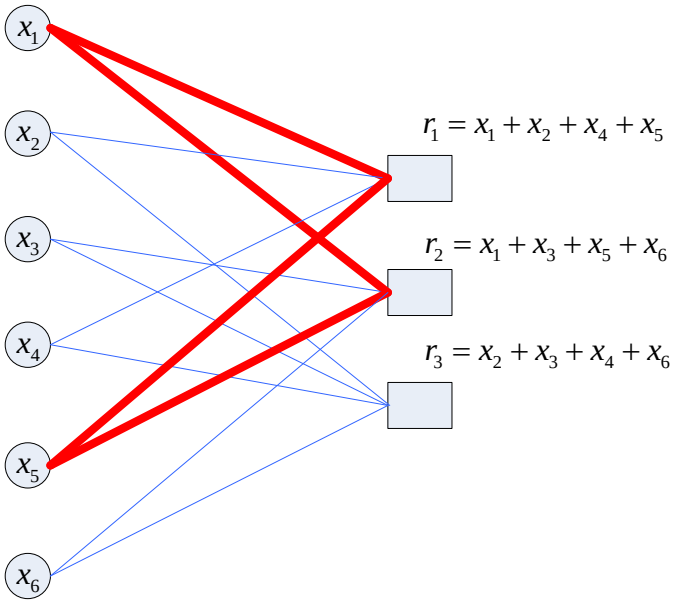


图 2 Girth 值为 4 的短循环

对应于 H 矩阵，此 4 循环可以表达为：

$$\mathbf{H}=\begin{bmatrix}1 & 1 & 0 & 1 & 1 & 0\\ 1 & 0 & 1 & 0 & 1 & 1\\ 0 & 1 & 1 & 1 & 0 & 1\end{bmatrix}$$

同理，对于 6 循环我们有：

$$\mathbf{H}_1=\begin{bmatrix}1 & 0 & 1\\ 1 & 1 & 0\\ 0 & 1 & 1\end{bmatrix}$$

上式所示的矩阵 $\mathbf{H}_1$ 表示校验矩阵 H 中的一个子矩阵。

短循环对于矩阵性能的影响： 在 LDPC 码的译码算法中，我们都假设传递的消息满足彼此独立的假设。当 H 矩阵中存在长度为 2*L* 的环路时，则这些消息只在前 *L* 轮迭代过程中满足独立性假设（注意：解码的迭代过程包括一次比特节点更新和一次校验节点更新）。因此短循环的存在会影响 LDPC 码的解码性能。

另外简单的解释：例如在图 2 所示的 H 矩阵中，存在一个 4 循环于是这个 4 循环对应的校验方程为：

$$r_1 = x_1 + x_2 + x_4 + x_5$$

$$r_2 = x_1 + x_3 + x_5 + x_6$$

由于 4 循环的存在，我们可以看到，上面两个校验方程含有共同的比特节点 x_1 与 x_5，直观地说，如果这两个校验方程均出错，则我

们无法确定 x_1 与 x_5 中究竟哪个出错，所以从这个角度也可以说明短循环对于性能带来的影响。

8.4 LDPC 码的编码方法

LDPC 码属于线性分组码，利用其校验矩阵 H 可以生成编码矩阵 G，从而可以生成码字，其校验矩阵 H 可以体现 LDPC 码的特点与性能，所以我们首先介绍 H 矩阵的构造方法。

8.4.1 LDPC 码校验矩阵 H 的构造

Gallager 的 H 矩阵构造方法

- 每一行有 j 个 1；——满足行重要求
- 每一列有 k 个 1；——满足列重要求
- 任意两列具有共同 1 的个数不大于 1；——满足 girth 值大于 4
- j 和 k 分别与 H 矩阵中的列数和行数相比小得多；——满足 H 矩阵的稀疏性

设矩阵 H_0 为：

$$H_0 = \begin{bmatrix} \underbrace{11\ldots1}_{j} & & \\ & \underbrace{11\ldots1}_{j} & \\ & \ddots & \\ & & \underbrace{11\ldots1}_{j} \end{bmatrix}$$

利用 H_0，我们可以通过列交换的方法得到校验矩阵 H：

$$\mathbf{H} = \begin{bmatrix} \pi_1(H_0) \\ \pi_2(H_0) \\ \vdots \\ \pi_k(H_0) \end{bmatrix}$$

例如下图就是由这种方法构造出的校验矩阵：

$$
\mathbf{H} = \left[\begin{array}{cccccccccccccccccc}
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1\\
\hdashline
1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0\\
0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&0\\
0&0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&0\\
0&0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0\\
0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1\\
\hdashline
1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0\\
0&1&0&0&0&0&1&0&0&0&1&0&0&0&0&1&0&0&0&0\\
0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&1&0\\
0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&1&0&0&0\\
0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1
\end{array}\right]
$$

Mackay 的构造方法

此方法中，校验矩阵的列重为 *j*，每行的平均重量为 *k*（即此方法所构造出的 H 矩阵），且任意两列具有共同 1 的个数不大于 1（保证不存在 4 循环）。

构造法 1A

这是一种基本的构造方法，每一列具有固定的列重 *j*。随机构造矩阵，使其平均行重为 *k*，且任意两列具有共同 1 的个数不大于 1。构造矩阵如图 3 所示：

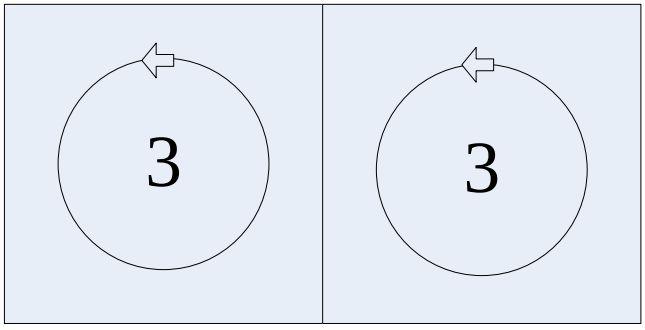


图 3（*j*=3，*k*=6，*R*=1/2）

可以按照上面的方法生成一个 H 矩阵，例如

$$\left[\begin{array}{cccccc|cccccc} 1&0&0&0&0&0&1&1&1&1&1&0\\ 1&1&0&1&0&1&1&1&0&0&0&0\\ 0&1&1&1&1&0&0&0&1&0&0&1\\ 1&1&0&0&1&1&1&0&0&1&0&0\\ 0&0&1&1&0&1&0&0&1&0&1&1\\ 0&0&1&0&1&0&0&1&0&1&1&1 \end{array}\right]$$

构造法 2A

与构造法 1A 类似，引入一些列重为 2 的列，使得 H 矩阵的 girth 值增大，其中列重为 2 的列数为 $m/2$，这一部分是由两个 $m/2\times m/2$ 的单位阵上下重叠起来构成的。如下图所示：

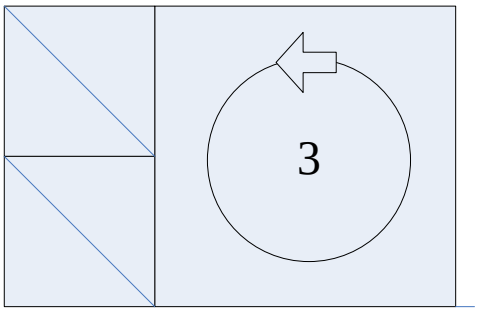


图 4

可以按照上面的方法生成一个 H 矩阵，例如

$$\left[\begin{array}{ccc|ccccccccc} 1&0&0&0&0&0&1&1&1&1&1&0\\ 0&1&0&1&0&1&1&1&0&0&0&0\\ 0&0&1&1&1&0&0&0&1&0&0&1\\ 1&0&0&0&1&1&1&0&0&1&0&0\\ 0&1&0&1&0&1&0&0&1&0&1&1\\ 0&0&1&0&1&0&0&1&0&1&1&1 \end{array}\right]$$

构造法 1B 和 2B

从构造法 1A 和 2A 中删除一些仔细选择的列，使得 H 矩阵中 girth 值满足相应的要求。

超轻矩阵

将 Mackay 的构造方法推广，进一步增加列重为 2 的列数，即用更多的更小的单位矩阵连续重叠。现将两个 $m/2\times m/2$ 的单位矩阵重叠，再是 $m/4\times m/4$ 的单位矩阵重叠，以此类推，最终最多是 m 个列重为 2 的列。如下图所示。

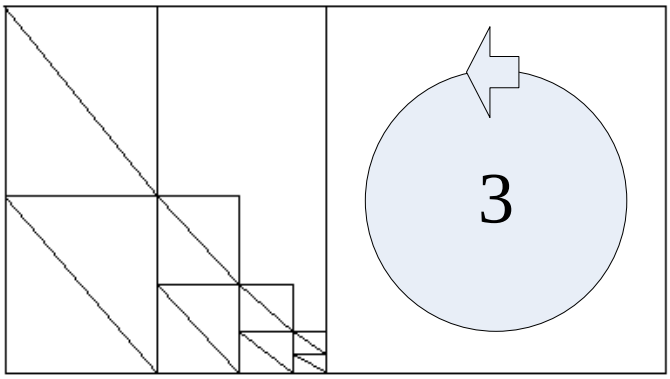


图 5

例如，

$$\left[\begin{array}{cccccc|cccccc} 1&0&0&0&0&0&1&1&1&1&1&0\\ 0&1&0&0&0&0&1&1&0&0&0&0\\ 0&0&1&0&0&0&0&0&1&0&0&1\\ \hline 1&0&0&1&0&0&1&0&0&1&0&0\\ 0&1&0&0&1&0&0&0&1&0&1&1\\ 0&0&1&1&0&1&0&1&0&1&1&1 \end{array}\right]$$

8.4.2 QC-LDPC 码的构造

在构造 LDPC 码校验矩阵时，随机构造方法不利于硬件实现，于是人们想到利用几何代数的方法来构造 LDPC 码的校验矩阵，于是就

产生了 QC-LDPC 码，即准循环 LDPC 码（Quasi-Cyclic LDPC codes）。首先，这类码有严谨的数学结构，构造和性能分析更加精确，甚至最小汉明距离都是可以计算的；其次，和随机构造的 LDPC 码相比，它具有更低的误码平台；第三，这类码字具有准循环结构，极大地降低了编码复杂度，也为译码提供了更方便的选择。

QC-LDPC 码的数学基础

首先，介绍 QC-LDPC 码的数学基础，以及重要的定理。

- 群：群(G)是具有二元运算的集合，并且适合一下条件：
  - 结合律成立，即 $(ab)c = a(bc), \forall a,b,c \in G$；
  - G 中存在一个元 ***e***： $ea = ae = a, \forall a \in G$；
  - 对 G 中的任意元 ***a***，存在 $a^{-1} \in G$，使 $aa^{-1} = a^{-1}a = e$。
- 环：有两个二元运算（分别叫做加法和乘法）的代数系统 $(A,+,\cdot)$ 叫做一个环，如果
  - $(A,+)$ 是一个加法群；
  - $(A,\cdot)$ 是一个半群（只对结合律成立）；
  - 乘法对加法的左、右分配律都成立。
- 域：一个交换的除环叫做一个域

一个重要的定理：（费马定理）

设 ***p*** 为素数，则有：$a^{p-1}=1 \pmod p$，即 $a^{p}=a \pmod p$。□

QC-LDPC 码的定义

定义 H 矩阵的 $m \times n$ 的母矩阵 M(H)，将 M(H)中的 0 与 1 分别用 L*L 的全 0 子矩阵与 L*L 的循环子矩阵 $\mathbf{p}^{a_{ij}}$ 替换，就可得到校验矩阵 H，

其中循环移位子矩阵 P 定义为

$$\mathbf{P}=\begin{bmatrix}0 & 1 & 0 & \cdots & 0\\ 0 & 0 & 1 & \cdots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \cdots & 1\\ 1 & 0 & 0 & \cdots & 0\end{bmatrix}$$

例如定义一个全 1 母矩阵为：

$$\mathbf{M(H)}=\begin{bmatrix}1 & 1 & \cdots & 1 & 1\\ 1 & 1 & \cdots & 1 & 1\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ 1 & 1 & \cdots & 1 & 1\end{bmatrix}_{m\times n}$$

将上式中的每个 1 替换成一个 $L\times L$ 的子矩阵 $\mathbf{p}^{a_{ij}}$，得到一个 $mL\times nL$ 的 H 矩阵：

$$\mathbf{H}=\begin{bmatrix}\mathbf{p}^{a_{11}} & \mathbf{p}^{a_{12}} & \cdots & \mathbf{p}^{a_{1(n-1)}} & \mathbf{p}^{a_{1n}}\\ \mathbf{p}^{a_{21}} & \mathbf{p}^{a_{22}} & \cdots & \mathbf{p}^{a_{2(n-1)}} & \mathbf{p}^{a_{2n}}\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ \mathbf{p}^{a_{m1}} & \mathbf{p}^{a_{m2}} & \cdots & \mathbf{p}^{a_{m(n-1)}} & \mathbf{p}^{a_{mn}}\end{bmatrix}$$

其中 a_{ij} (i=1,2, ⋯, m; j=1,2, ⋯,n)为移位项。将单位阵向右循环移位 a_{ij} 得到循环子矩阵 $\mathbf{p}^{a_{ij}}$。在存储 H 矩阵的时候，我们只需要存储上式中每一个 a_{ij} 的值，而不需要存储每个 1 的位置。

基于有限几何（Finite Geometries）的 LDPC 码的构造方法

有限几何指的是一个由点和边组成的集合，a 为其中一个 m 元组

点，即 $a \in GF(q)^m$ 。

在构造 LDPC 码的时候，有限几何 *G* 是由 *n* 个点和 *m* 条边组成，*n* 和 *m* 分别对应于 H 矩阵的列数和行数。我们要构造规则的 H 矩阵，所以这些点和边具有下列性质：

- 每一条边含有 *j* 个点——对应 H 矩阵的行重
- 每一个点都位于 *k* 条边上——对应 H 矩阵的列重
- 任意两个点仅仅通过一条边相连——确保没有 4 循环
- 任意两条边或平行或只相交于一点

对于这样的有限几何 *G*，所构造 H 矩阵的行和列分别对应其中的边和点，当 H 矩阵中 $h_{i,j}$=1 时，表示 *G* 中的第 *i* 条边包含了第 *j* 个点。

基于有限环的构造方法

有限环构造的优点：

由定义可知，环与域的区别在于除法和对于乘法的交换律。因此环的限制比域小得多。更重要的是环的基数可以做到连续，这样使得所构造的码长可以连续。

基于这种理论我们可以构造出 girth 值为 10，且码长连续的 LDPC 码 H 矩阵。

8.4.3 LDPC 码的线性编码

基本编码方法

- 通过矩阵的初等变换，把 H 矩阵变化成系统形式：$\mathbf{H}' = [\mathbf{P}^{\mathrm{T}}, \mathbf{I}]$；
- 得到该 H 矩阵对应的生成矩阵 $\mathbf{G} = [\mathbf{I}, \mathbf{P}]$；
- 用信息比特去乘生成矩阵 G，就得到了编码后的码字；

- 此方法的复杂度为$O(n^2)$。

Richardson 的 LDPC 码编码算法

1) 公式推导：

通过行列置换将 **H** 矩阵化为近似下三角形状，即：

$$\mathbf{H}=\begin{pmatrix}\mathbf{A}_{(m-g)\times(n-m)} & \mathbf{B}_{(m-g)\times g} & \mathbf{T}_{(m-g)\times(m-g)}\\ \mathbf{C}_{g\times(n-m)} & \mathbf{D}_{g\times g} & \mathbf{E}_{g\times(m-g)}\end{pmatrix}_{m\times n},$$

其中 **T** 为下三角阵。

即 **H** 可以转化为：

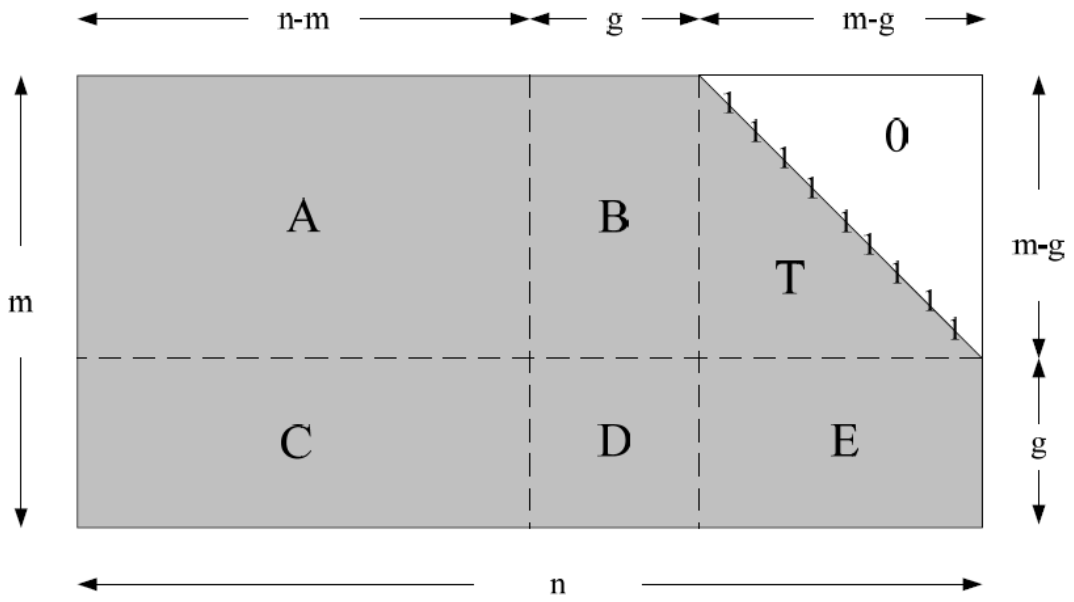


图 **6** 校验矩阵转换为近似下三角形式

其中 **g** 称为近似表示的 **gap**，**g** 越小，编码复杂度越低。

举例：假设$H=\begin{bmatrix}1&1&0&1&1&0&0&1&0&0\\0&1&1&0&1&1&1&0&0&0\\0&0&0&1&0&0&0&1&1&1\\1&1&0&0&0&1&1&0&1&0\\0&0&1&0&0&1&0&1&0&1\end{bmatrix}$，信息序列 **u=[1 1 0 0 1]**，

求编码序列。

注意这是一个编码速率 $r=1-\frac{2*\frac{7}{10}+3*\frac{3}{10}}{4*\frac{2}{5}+5*\frac{3}{5}}=\frac{1}{2}$ 的 LDPC 码，先对 H 矩阵进行行列置换（交换第 2 行、第 3 行，交换第 6 列、第 10 列），得到近似下三角形式。

$$H_t=\left[\begin{array}{cccccc|cc|ccc}1&1&0&1&1&0&0&1&0&0\\0&0&0&1&0&1&0&1&1&0\\0&1&1&0&1&0&1&0&0&1\\\hline1&1&0&0&0&0&1&0&1&1\\0&0&1&0&0&1&0&1&0&1\end{array}\right]\quad H_t=\left[\begin{array}{cccccc|cc|ccc}1&1&0&1&1&0&0&1&0&0\\0&0&\mathbf{A}&1&0&\mathbf{B}&0&1&\mathbf{T}&0\\0&1&1&0&1&0&1&0&0&1\\\hline1&1&\mathbf{C}&0&0&\mathbf{D}&1&0&\mathbf{E}&1\\0&0&1&0&0&1&0&1&0&1\end{array}\right]$$

应用 Gauss-Jordan 消除法把 E 消除（让 E=0），这就是让 $\begin{pmatrix}\mathbf{I}_{m-g}&\mathbf{0}\\\mathbf{-ET^{-1}}&\mathbf{I}_g\end{pmatrix}$ 矩阵左乘 Ht，即

$$\tilde{H}=\begin{bmatrix}\mathbf{I}_{m-g}&\mathbf{0}\\\mathbf{-ET^{-1}}&\mathbf{I}_g\end{bmatrix}H_t=\begin{bmatrix}\mathbf{I}_{m-g}&\mathbf{0}\\\mathbf{-ET^{-1}}&\mathbf{I}_g\end{bmatrix}\begin{bmatrix}A&B&T\\C&D&E\end{bmatrix}$$
$$=\begin{bmatrix}A&B&T\\\tilde{C}&\tilde{D}&\mathbf{0}\end{bmatrix}$$

其中 $\tilde{C}=-ET^{-1}A+C$、$\tilde{D}=-ET^{-1}B+D$、$\tilde{E}=-ET^{-1}T+E=\mathbf{0}$

$$T^{-1}=\begin{bmatrix}1&0&0\\1&1&0\\0&0&1\end{bmatrix},\quad\begin{bmatrix}\mathbf{I}_{m-g}&\mathbf{0}\\\mathbf{-ET^{-1}}&\mathbf{I}_g\end{bmatrix}=\begin{bmatrix}1&0&0&0&0\\0&1&0&0&0\\0&0&1&0&0\\1&1&1&1&0\\1&0&1&0&1\end{bmatrix}$$

$$\tilde{H}=\left[\begin{array}{cccccc|cc|ccc}1&1&0&1&1&0&0&1&0&0\\0&0&0&1&0&1&0&1&1&0\\0&1&1&0&1&0&1&0&0&1\\\hline0&1&1&0&0&1&0&0&0&0\\1&0&0&1&0&1&1&0&0&0\end{array}\right]$$

假设编码的码字为 $\mathbf{c}=[c_1,c_2,\ldots,c_N]$ 分为三部分 $\mathbf{c}=[\mathbf{u}\ \ \mathbf{p}^{(1)}\ \ \mathbf{p}^{(2)}]$，其中

$\mathbf{u}=[u_1,u_2,\ldots,u_K]$是 K 比特信息序列，$\mathbf{p}^{(1)}=\left[p_1^{(1)},p_2^{(1)}\cdots p_g^{(1)}\right]$是第一部分校验比特，$\mathbf{p}^{(2)}=\left[p_1^{(2)},p_2^{(2)}\cdots p_{m-g}^{(2)}\right]$是第二部分校验比特。

因为$\mathbf{c}\tilde{H}^T=\mathbf{0}$，所以有：

$$\mathbf{u}A^T+\mathbf{p}^{(1)}B^T+\mathbf{p}^{(2)}T^T=\mathbf{0}$$

$$\mathbf{u}\tilde{C}^T+\mathbf{p}^{(1)}\tilde{D}^T=\mathbf{0}$$

所以，$\mathbf{p}^{(1)}=\mathbf{u}\tilde{C}^T\left(\tilde{D}^T\right)^{-1}$、$\mathbf{p}^{(2)}=\left(\mathbf{u}A^T+\mathbf{p}^{(1)}B^T\right)\left(T^T\right)^{-1}$

本例中，

$$\mathbf{p}^{(1)}=\mathbf{u}\tilde{C}^T\left(\tilde{D}^T\right)^{-1}=[1\quad 1\quad 0\quad 0\quad 1]\begin{bmatrix}0&1\\1&0\\1&0\\0&1\\0&0\end{bmatrix}\begin{bmatrix}1&1\\0&1\end{bmatrix}=[1\quad 1]\begin{bmatrix}1&1\\0&1\end{bmatrix}=[1\quad 0]$$

$$\mathbf{p}^{(2)}=\left(\mathbf{u}A^T+\mathbf{p}^{(1)}B^T\right)\left(T^T\right)^{-1}$$

$$=\left(\begin{matrix}\\[1\quad 1\quad 0\quad 0\quad 1]\begin{bmatrix}1&0&0\\1&0&1\\0&0&1\\1&1&0\\1&0&1\end{bmatrix}+[1\quad 0]\begin{bmatrix}0&1&0\\0&0&1\end{bmatrix}\\\\\end{matrix}\right)\left(T^T\right)^{-1}$$

$$=[1\quad 1\quad 0]\begin{bmatrix}1&1&0\\0&1&0\\0&0&1\end{bmatrix}=[1\quad 0\quad 0]$$

所以，整个编码序列为 c=[1 1 0 0 1 1 0 1 0 0]

8.5 通过 H 矩阵的设计简化编码复杂度

在利用 LDPC 码 H 矩阵编码的时候，一般都需要先得到 G 矩阵，随后生成所需要的码字。在 H 矩阵转化为 G 矩阵的过程中所需要的计算量是相当大的。而且在硬件存储方面，H 矩阵为稀疏矩阵，需要

存储每个 1 的位置，但是 G 矩阵是一个密集矩阵加上一个单位阵的形式，这就需要开辟很大的存储空间。所以如果编码时不需要生成 G 矩阵，而直接通过 H 矩阵编码，这样就将编码复杂度大大简化。

最简单的方法

这种方法是构造 H_{sys} 形式的 H 矩阵。即先构造一个矩阵 H1，随后在 H1 的后面添加一个单位阵构造出 H 矩阵。即：

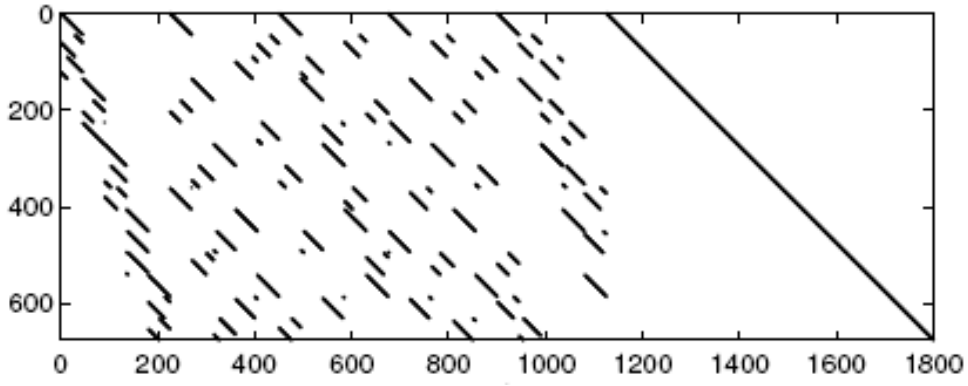


图 7

利用这种 H 矩阵可以实现线性时间编码。

递推编码的方法

由于上面的方法最右边存在一个单位阵，导致最右边列重为 1，即最右边每个比特只参与一个校验方程。这样使得校验节点对比特节点的保护就不够，造成性能上的降低。于是为了增加对右边比特的保护，我们可以采用下面 H 矩阵构造方式。

将 H 矩阵构造为如下形式：

$$H=\begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,(n-m)} & p_{1,(n-m+1)} & 0 & 0 & \cdots & 0 \\ p_{2,1} & p_{2,2} & \cdots & p_{2,(n-m)} & p_{2,(n-m+1)} & p_{2,(n-m+2)} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{m,1} & p_{m,2} & \cdots & p_{m,(n-m)} & p_{m,(n-m+1)} & p_{m,(n-m+2)} & p_{m,(n-m+3)} & \cdots & p_{m,n} \end{bmatrix}$$

其中 $p_{i,j}$ 为利用单位阵移位生成的循环子矩阵。这种循环子矩阵有一个非常好的性质：

$p_{i,j}p_{i,j}^T=I$；即 $p_{i,j}^{-1}=p_{i,j}^T$。这样我们就可以通过对矩阵求转置而得到矩阵的逆，大大简化了计算量。

对于上式所示的 H 矩阵，由其校验方程 $Hx=0$，可得：

$$\begin{cases} p_{1,(n-m+1)}c_1 = p_{1,1}x_1 + p_{1,2}x_2 + \cdots + p_{1,(n-m)}x_{n-m} \\ p_{2,(n-m+1)}c_1 + p_{2,(n-m+2)}c_2 = p_{2,1}x_1 + p_{2,2}x_2 + \cdots + p_{2,(n-m)}x_{n-m} \\ \vdots \\ p_{m,(n-m+1)}c_1 + p_{m,(n-m+2)}c_2 + \cdots + p_{m,n}c_m = p_{m,1}x_1 + p_{m,2}x_2 + \cdots + p_{m,(n-m)}x_{n-m} \end{cases};$$

则

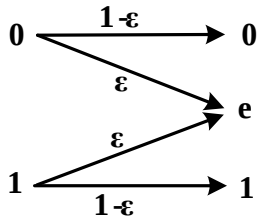
$$\begin{cases} c_1 = p_{1,(n-m+1)}^T(p_{1,1}x_1 + p_{1,2}x_2 + \cdots + p_{1,(n-m)}x_{n-m}) \\ c_2 = p_{2,(n-m+2)}^T(p_{2,1}x_1 + p_{2,2}x_2 + \cdots + p_{2,(n-m)}x_{n-m} + p_{2,(n-m+1)}c_1) \\ \vdots \\ c_m = p_{m,n}^T(p_{m,1}x_1 + p_{m,2}x_2 + \cdots + p_{m,(n-m)}x_{n-m} + p_{m,(n-m+1)}c_1 + p_{m,(n-m+2)}c_2 + \cdots) \end{cases};$$

其中 $x = \begin{bmatrix} x_1 \\ \vdots \\ x_{n-m} \\ c_1 \\ \vdots \\ c_m \end{bmatrix}$，利用递推算法可以得到相应的 $c_1, c_2, \cdots, c_m$。

8.6 LDPC 码的译码

译码算法我们主要讲述消息传递（Message-Passing）算法、比特翻转（Bit-Flipping）译码算法、和积（Sum-Product）算法。

8.6.1 BEC 信道（Binary Erasure Channel）下的消息传递算法



在 BEC 信道中，接收端要么正确接收（概率为$1-\varepsilon$），要么接收到一个消息 e（概率为ε），其实就表明这个比特没有正确接收，被擦除了。消息传递算法是一种迭代译码算法，通过比特节点和校验节点间的前向和后向消息的传递，直到得到一个结果（迭代次数达到或校验方程都满足即 $cH^T=0$）。

例：假设一个码 $C=[c_1\ c_2\ c_3\ c_4\ c_5\ c_6]$满足以下校验方程：

$c_1+c_2+c_4=0$

$c_1+c_2+c_3+c_6=0$

$c_2+c_3+c_5=0$

每个合法有效的码字都必然遵守上述的约束。如果收到一个码序列为 [1 1 0 0 0 0]，通过演算可知：

$c_1+c_2+c_4=1+1+0=0$

$c_1+c_2+c_3+c_6=1+1+0+0=0$

$c_2+c_3+c_5=1+0+0=1$

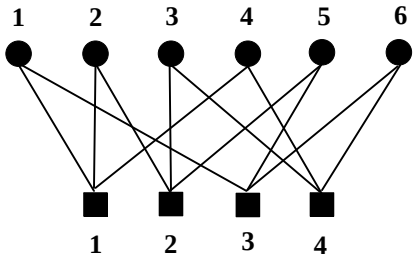
不满足其中一个约束方程，所以上述码字不是一个合法有效的码字。对于第一个约束方程，$c_1+c_2+c_4=0$，如果已知 $c_1=0$，$c_2=1$，那么我们可以计算得到 c_4 必然为 1，即 $c_4=1$，这就是消息传递译码的基础，每个校验节点能够决定擦除比特的值（当然，前提是校验方程中只有一个擦除比特，如果有多个擦除比特，就求出不来了）。

第 i 个比特节点发送信息 M_i 给它连接的校验节点，要么值是已知的（1 或 0），要么就是 e（表明它被擦除了）。

第 j 个校验节点发送给与它相连的第 i 个比特节点的信息 $E_{j,i}$，阐明 第 i 个比特是 1、0 或 e，这样一个迭代过程就完成了。直到所有比特节点的值都已知，或达到了最大迭代次数，译码结束。

我们用 B_j 来表示 H 中第 j 个校验方程中比特集合，比如：

$$H=\begin{bmatrix}1&1&0&1&0&0\\0&1&1&0&1&0\\1&0&0&0&1&1\\0&0&1&1&0&1\end{bmatrix},$$



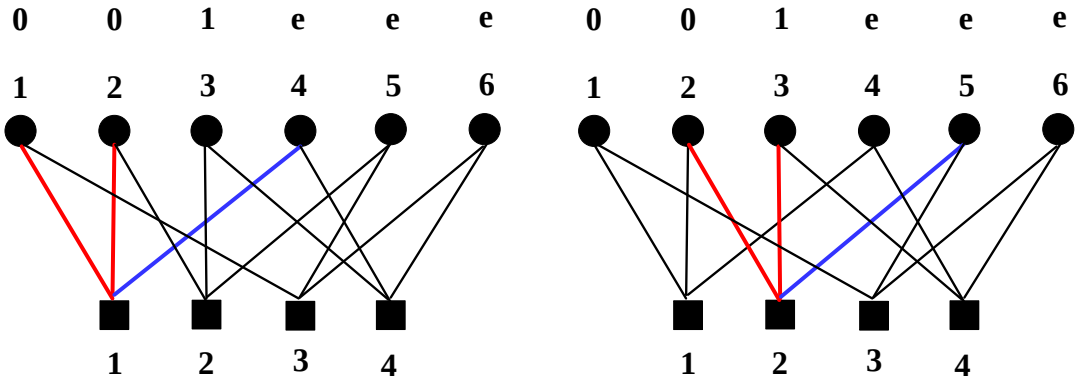
B_1={1,2,4}，B_2={2,3,5}，B_3={1,5,6}，B_4={3,4,6}

类似地，我们用 A_i 表示第 i 个比特参与的校验方程集合，

A_1={1,3}，A_2={1,2}，A_3={2,4}，A_4={1,4}，A_5={2,3}，A_6={3,4}

例：码字 c=[0 0 1 0 1 1]经过信道后，接收到的向量为 y=[0 0 1 e e e]，应用消息传递译码算法恢复出擦除比特。

初始阶段，$M_i=y_i$，所以 M=[0 0 1 e e e]

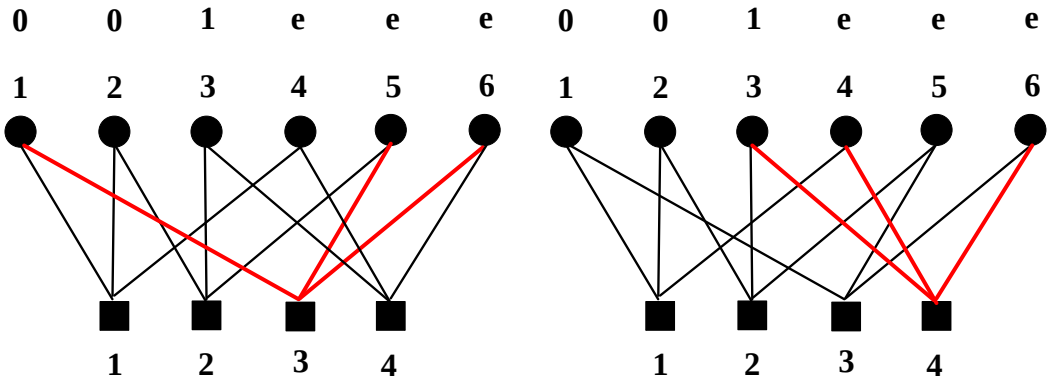


算法的第一步：通过校验节点计算某些比特信息。

第一个校验节点涉及到第 1、2 和 4 比特节点，相应的输入信息为 0 0 e，因为这个校验节点只有一个输入 e 信息（从第 4 个比特节点发出的），因此可以计算出这个比特的值 $E_{1,2}$，并将结果通过边（第 1 个校验节点到第 4 个比特节点的边）传递到第 4 个比特节点。

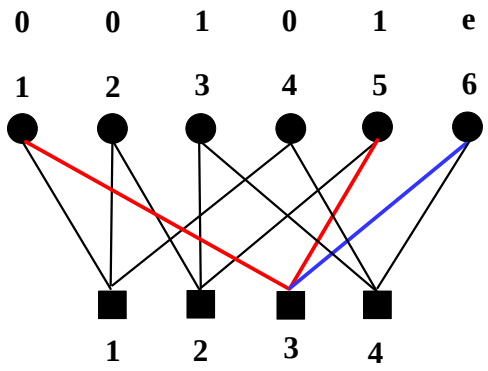
$E_{1,4}=M_1+M_2=0+0=0$

类似地，$E_{2,5}=M_2+M_3=0+1=1$，



但随后的第 3 个校验节点是由第 1、5、6 比特节点参与的，即 0、e、e，因为有两个 e，所以无法确定任何比特的值；第 4 个校验节点也是类似的。

第二步：比特信息更新，M=[0 0 1 0 1 e]



$E_{3,6}=M_1+M_5=0+1=1$，

同时，也满足第 4 个校验方程：$M_3+M_4+M_6=1+0+1=0$

这样，第 6 个比特节点为 1

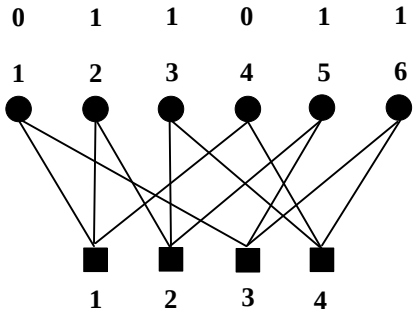
所有比特节点都计算出来，$\hat{c}=M=[0\ 0\ 1\ 0\ 1\ 1]$

8.6.2 比特翻转译码

它是对 LDPC 码进行硬判决译码，沿着 Tanner 图的边传递的信息都是 0、1 这样的二进制信息。第 j 个校验节点决定第 i 个比特节点的值：先假设第 i 个比特节点被擦除了，看看是 1 还是 0 满足第 j 个校验方程，这就称为第 i 个比特的外部信息。但注意：这里其他比特节点给校验节点的值不一定是正确的，这与前面 BEC 信道的情况是不同的。比特节点会收到多个外部信息，然后按照“少数服从多数”原则，得到该比特的值。这个过程一直重复，直到最大译码迭代次数达到或满足 $\hat{c}\cdot H^T=0$。

举例：假设发送的码字是 c=[0 0 1 0 1 1]，经过信道后，接收到的码序列是：y=[0 1 1 0 1 1]

初始阶段，$M_i=y_i$，所以 M=[0 1 1 0 1 1]

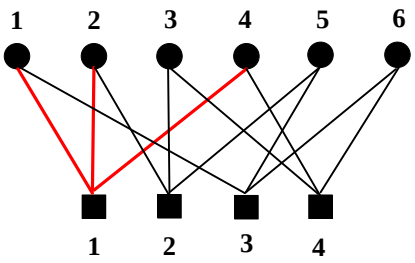


$B_1=\{1,2,4\}$

$E_{1,1}=M_2+M_4=1+0=1$，$E_{1,2}=M_1+M_4=0+0=0$，$E_{1,4}=M_1+M_2=0+1=1$

第1个校验反馈的值 1 0 1

接收到的值 0 1 1 0 1 1

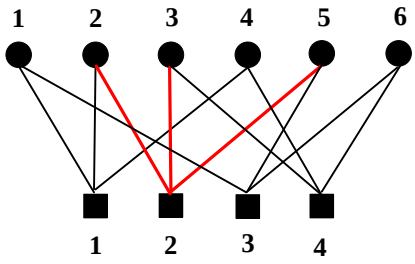


$E_{2,2}=M_3+M_5=1+1=0$，$E_{2,3}=M_2+M_5=1+1=0$，$E_{2,5}=M_2+M_3=1+1=0$

第2个校验反馈的值 0 0 0

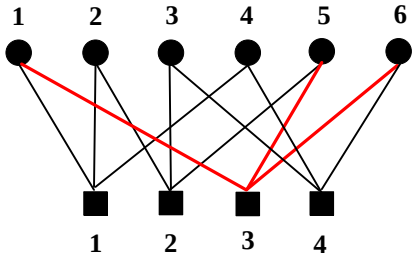
第1个校验反馈的值 1 0 1

接收到的值 0 1 1 0 1 1



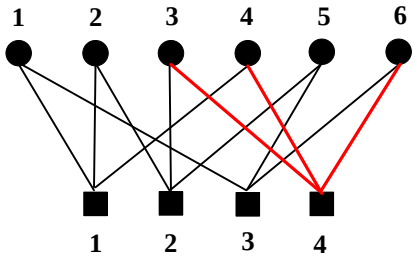
$E_{3,1}=M_5+M_6=1+1=0$，$E_{3,5}=M_1+M_6=0+1=1$，$E_{3,6}=M_1+M_5=0+1=1$

第3个校验反馈的值	0				1	1
第2个校验反馈的值		0	0		0	
第1个校验反馈的值	1	0		1		
接收到的值	0	1	1	0	1	1
	1	2	3	4	5	6



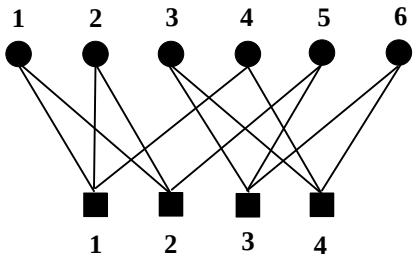
$E_{4,3}=M_4+M_6=0+1=1$，$E_{4,4}=M_3+M_6=1+1=0$，$E_{4,6}=M_3+M_4=1+0=1$

第4个校验反馈的值			1	0		1
第3个校验反馈的值	0				1	1
第2个校验反馈的值		0	0		0	
第1个校验反馈的值	1	0		1		
接收到的值	0	1	1	0	1	1
	1	2	3	4	5	6

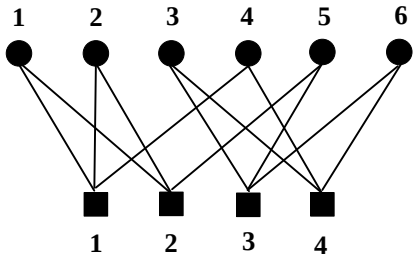


第 2 步：第 1 个比特节点，接收到校验节点 1 和 3 送来的外部信息，分别是 1 和 0，所以仍然保持原来的值 0；第 2 个比特节点，收到校验节点 1 和 2 送来的外部信息 0 和 0，和原来的值 1 不同，就对 1 进行翻转 1→0；以此类推，译码得到[0 0 1 0 1 1]，成功纠错。

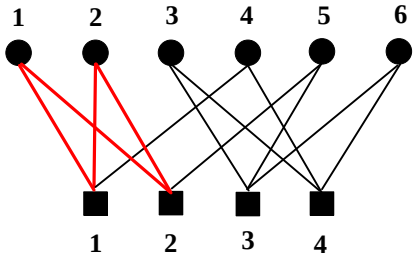
再举例：假设发送的码字是 c=[0 0 1 0 0 1]，经过信道后，接收到的码序列是：y=[1 0 1 1 0 1]



译码的值	0	1	1	0	0	1
第4个校验反馈的值			1	0		1
第3个校验反馈的值			1		0	1
第2个校验反馈的值	0	1			1	
第1个校验反馈的值	0	1		1		
接收到的值	1	0	1	0	0	1



再次计算，仍然得不到正确结果。



因为存在 4 循环，前两个比特参与两个校验方程，无法确定是哪个比特出错。

6.8.3 和积算法

和积算法是软判决消息传递算法，与前面的比特翻转算法类似（只是为 0 或 1 的概率，不是二进制的 0 或 1 了）。比特翻转算法的输入是 0、1（即对接收值做了硬判决），而和积算法的输入是 0 或 1 的概率（这是一个软信息），即每个接收值的先验概率。

对于和积译码器，节点间传递的外部信息也是概率值，而不是硬判决值。校验节点 j 到比特节点 i 的外部信息 $E_{j,i}$ 表示校验节点 j 认为 ci=1 的概率，即 $E_{j,i}$ 给出了“若 c_i=1，则校验方程 j 成立”的概率，该概率就是校验方程中有奇数个 1 的概率：

$$P_{j,i}^{ext}=\frac{1}{2}-\frac{1}{2}\prod_{i'\in B_j,i'\neq i}\left(1-2P_{j,i'}\right)$$

其中 $P_{j,i'}$ 是给第 j 个校验节点 $c_{i'}$=1 的概率。

这样，“若 c_i=0，则校验方程 j 成立”的概率就为 $1-P_{j,i}^{ext}$

对一个二进制变量 x，p(x=0)+p(x=1)=1，因此对变量 x 我们只需存储一个概率值，其对数表示为：

$$L(x)=\log\frac{p(x=0)}{p(x=1)}$$

这样，$p(x=0)=\frac{e^{L(x)}}{1+e^{L(x)}}$，$p(x=1)=\frac{e^{-L(x)}}{1+e^{-L(x)}}$，

对数处理能够减少计算复杂度，因此，Ej,i 表示为对数似然比形式，即：

$$E_{j,i}=L(P_{j,i}^{ext})=\log\frac{1-P_{j,i}^{ext}}{P_{j,i}^{ext}}$$

把前面的式子带进来，可得：

$$E_{j,i}=\log\frac{\frac{1}{2}+\frac{1}{2}\prod_{i'\in B_j,i'\neq i}\left(1-2P_{j,i'}\right)}{\frac{1}{2}-\frac{1}{2}\prod_{i'\in B_j,i'\neq i}\left(1-2P_{j,i'}\right)}=\log\frac{1+\prod_{i'\in B_j,i'\neq i}\left(1-2\frac{e^{-M_{j,i'}}}{1+e^{-M_{j,i'}}}\right)}{1-\prod_{i'\in B_j,i'\neq i}\left(1-2\frac{e^{-M_{j,i'}}}{1+e^{-M_{j,i'}}}\right)}$$

$$=\log\frac{1+\prod_{i'\in B_j,i'\neq i}\left(\frac{1-e^{-M_{j,i'}}}{1+e^{-M_{j,i'}}}\right)}{1-\prod_{i'\in B_j,i'\neq i}\left(\frac{1-e^{-M_{j,i'}}}{1+e^{-M_{j,i'}}}\right)}$$

其中 $M_{j,i'} \triangleq L(P_{j,i'}) = \log \frac{1-P_{j,i'}}{P_{j,i'}}$

根据数学表达式： $\tanh \frac{1}{2} \log\left(\frac{1-p}{p}\right) = 1-2p$ ，可得：

$$E_{j,i} = \log \frac{1+\prod_{i' \in B_j, i' \neq i} \tanh\left(M_{j,i'}/2\right)}{1-\prod_{i' \in B_j, i' \neq i} \tanh\left(M_{j,i'}/2\right)}$$

再利用数学公式 $2\tanh^{-1} p = \log \frac{1+p}{1-p}$ ，可得：

$$E_{j,i} = 2\tanh^{-1} \prod_{i' \in B_j, i' \neq i} \tanh\left(M_{j,i'}/2\right)$$

这样 ，第 i 个比特节点收到的总信息包括：该比特节点的输入 LLR---Ri、以及从与它相连的每个校验节点的外信息 Ej,i，即

$$L_i = R_i + \sum_{j \in A_i} E_{j,i}$$

其中 $R_i = \log \frac{p(c_i = 0 \mid y_i)}{p(c_i = 1 \mid y_i)}$

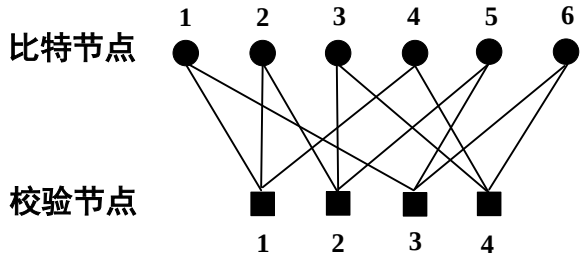
但是从比特节点 i 到校验节点 j 的信息 Mj,i 并不是上式，还要从上式中除去第 j 个校验节点发送的信息，即：

$$M_{j,i} = R_i + \sum_{\substack{j' \in A_i \\ j' \neq j}} E_{j',i}$$

和积算法旨在：（a）对每个码字比特计算出后验概率，（b）为每个比特选择具有最大后验概率的译码值。

举例：假设发送的码字是 c=[0 0 1 0 1 1]，经过 BSC 信道（ ε= 0.2 ）后，接收到的码序列是：y=[1 0 1 0 1 1]

$$H=\begin{bmatrix}1&1&0&1&0&0\\0&1&1&0&1&0\\1&0&0&0&1&1\\0&0&1&1&0&1\end{bmatrix},$$



$$R_i=\begin{cases}\log\dfrac{\varepsilon}{1-\varepsilon}, & y_i=1\\ \log\dfrac{1-\varepsilon}{\varepsilon}, & y_i=0\end{cases},$$

对该 **BSC** 信道而言，$\log\frac{\varepsilon}{1-\varepsilon}=\log\frac{0.2}{0.8}=-1.3863$，$\log\frac{1-\varepsilon}{\varepsilon}=\log\frac{0.8}{0.2}=1.3863$，

因此 **R=[-1.3863, 1.3863, -1.3863, 1.3863, -1.3863, -1.3863]**

假设最大迭代次数为 **3**，初始化时，$M_{j,i}=R_i$

第 **1** 个比特涉及第 **1** 和第 **3** 个校验方程，$M_{1,1}=R_1=-1.3863$, $M_{3,1}=R_1=-1.3863$

其他比特也是类似，

i=2，$M_{1,2}=R_2=1.3863$, $M_{2,2}=R_2=1.3863$

i=3，$M_{2,3}=R_3=-1.3863$, $M_{4,3}=R_3=-1.3863$

i=4，$M_{1,4}=R_4=1.3863$, $M_{4,4}=R_4=1.3863$

i=5，$M_{2,5}=R_5=-1.3863$, $M_{5,5}=R_5=-1.3863$

i=6，$M_{3,6}=R_6=-1.3863$, $M_{4,6}=R_6=-1.3863$

现在计算校验节点到比特节点的外部概率信息，第 **1** 个校验包含第 **1**、第 **2** 和第 **4** 比特，所以从第 **1** 个校验节点到第 **1** 个比特节点的外部概率就取决于第 **2** 和第 **4** 比特的概率，即

$$E_{1,1}=\log\frac{1+\tanh\left(M_{1,2}/2\right)\tanh\left(M_{1,4}/2\right)}{1-\tanh\left(M_{1,2}/2\right)\tanh\left(M_{1,4}/2\right)}$$
$$=\log\frac{1+\tanh\left(1.3863/2\right)\tanh\left(1.3863/2\right)}{1-\tanh\left(1.3863/2\right)\tanh\left(1.3863/2\right)}$$
$$=\log\frac{1+0.6*0.6}{1-0.6*0.6}=0.7538$$

类似地，可求出 $E_{1,2}$=-0.7538，$E_{1,4}$=-0.7538

第 2 个校验节点连接着第 2、3、5 比特节点，因此外部的 LLR 为：

$$E_{2,2}=\log\frac{1+\tanh\left(M_{2,3}/2\right)\tanh\left(M_{2\,5}/2\right)}{1-\tanh\left(M_{2,3}/2\right)\tanh\left(M_{2\,5}/2\right)}$$
$$=\log\frac{1+\left(-0.6\right)*\left(-0.6\right)}{1-\left(-0.6\right)*\left(-0.6\right)}=0.7538$$

$E_{2,3}$=-0.7538，$E_{2,5}$=-0.7538

重复所有校验节点，可得外部 LLR 矩阵（为了节省空间，存储成矩阵形式）：

$$E=\begin{bmatrix}0.7538 & -0.7538 & \cdot & -0.7538 & \cdot & \cdot\\ \cdot & 0.7538 & -0.7538 & \cdot & -0.7538 & \cdot\\ 0.7538 & \cdot & \cdot & \cdot & 0.7538 & 0.7538\\ \cdot & \cdot & -0.7538 & 0.7538 & \cdot & -0.7538\end{bmatrix}$$

对每个比特计算估计的后验概率，硬判决后，检查校正子 s，就可知道是否为一个合法码字。

第 1 个比特节点收到第 1、第 3 校验节点发来的外部 LLR 值，还有一个从信道收到的内部 LLR 值，则总的 LLR 值为：

$L_1=R_1+E_{1,1}+E_{3,1}$=-1.3863+0.7538+0.7538=0.1213

这表明第 1 个比特译为 0

同理，$L_2=R_2+E_{1,2}+E_{2,2}$=1.3863

$L_3=R_3+E_{2,3}+E_{4,3}$=-2.8938

$L_4=R_4+E_{1,4}+E_{4,4}=1.3863$

$L_5=R_5+E_{2,5}+E_{3,5}=-1.3863$

$L_6=R_6+E_{3,6}+E_{4,6}=-1.3863$

硬判决译码后的序列为：$\hat{\mathbf{c}} = [0\ 0\ 1\ 0\ 1\ 1]$

用校正子 **s** 进行检验，看看是否为一个合法码字

$$s = \hat{\mathbf{c}}H^T = [0\ 0\ 1\ 0\ 1\ 1]\begin{bmatrix}1&0&1&0\\1&1&0&0\\0&1&0&1\\1&0&0&1\\0&1&1&0\\0&0&1&1\end{bmatrix} = \begin{bmatrix}0&0&0&0\end{bmatrix}$$

译码结束。

当然，还有一些计算复杂度低的改进方法，就不叙述了。